Acromag®

Series IP-EP201/202/203/204 Industrial I/O Pack Engineering Design Kit

FPGA PROGRAMMING GUIDE

ACROMAG INCORPORATED
30765 South Wixom Road
P.O. BOX 437
Wixom, MI 48393-7037 U.S.A.

Tel: (248) 295-0310
Fax: (248) 624-9234

Copyright 2011, Acromag, Inc., Printed in the USA.
Data and specifications are subject to change without notice.

8500-798-D13F007

TABLE OF CONTENTS

IMPORTANT SAFETY CONSIDERATIONS
You must consider the possible negative effects of power, wiring, component, sensor, or software failure in the design of any type of control or monitoring system. This is very important where property loss or human life is involved. It is important that you perform satisfactory overall system design and it is agreed between you and Acromag, that this is your responsibility.

1.0 GETTING STARTED

GENERAL INFORMATION..... 3
DESIGN FILES..... 3
LOADING THE PROJECT..... 4
PROJECT SETTINGS..... 4
IP-EP2 ASSIGNMENTS..... 8
REQUIRED VHDL LOGIC..... 9
COMPILATION..... 11

2.0 PROGRAMMING THE BOARD

FPGA CONFIGURATION OVER THE IP BUS..... 12
DIRECT FPGA CONFIGURATION VIA JTAG..... 13
FLASH CONFIGURATION VIA JTAG..... 14

3.0 HARDWARE PROGRAM DISABLE

DISABLE PROGRAMMING OVER THE IP BUS..... 17
DISABLE JTAG PROGRAMMING..... 17
REMOVING THE CONFIGURATION JUMPER..... 17

4.0 TROUBLESHOOTING

FREQUENTLY ASKED QUESTIONS..... 18

APPENDIX

PIN ASSIGNMENTS..... 21
WEAK PULL-UP ASSIGNMENTS..... 25
DEVICE SETTINGS..... 25
REVISION INFORMATION..... 26

The information of this manual may change without notice. Acromag makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Further, Acromag assumes no responsibility for any errors that may appear in this manual and makes no commitment to update, or keep current, the information contained in this manual. No part of this manual may be copied or reproduced in any form without the prior written consent of Acromag, Inc.

RELATED PUBLICATIONS

The following manuals and part specifications provide the necessary information for in depth understanding of the IP-EP2 Series board.

IP-EP2 Series User's Manual	**www.acromag.com**
71V016SA SRAM Specifications	**http://www.idt.com**
Cyclone II Data Book	**http://www.altera.com**
CY22150 Specification	**http://www.cypress.com**

Trademarks are the property of their respective owners.

1.0 GETTING STARTED

The Industrial I/O Pack EP2 Series module is a reconfigurable digital input/output board. The modules use an Altera Cyclone II® Field Programmable Gate Array (FPGA). This allows designers to implement logic functions unique to their application. Furthermore, the FPGA can be configured in-system using either the industry-standard JTAG interface or directly through the IP bus.

GENERAL INFORMATION

The IP-EP2 Series Engineering Design Kit contains an example Altera FPGA program, including configuration files and the corresponding VHDL source files. The example design includes an IP bus interface to ID space, IO space and Interrupt space. IO space is used to access a 64K x 16 RAM array, control field data I/O, and control a clock generation chip. This guide assumes that the user is proficient in the use of VHDL and the Altera Quartus II software tools.

Prior to editing any of the VHDL code, the user should become familiar with the example design, as provided by Acromag. *Do not attempt to reconfigure the FPGA until after you have thoroughly tested the IP-EP2 Series module and understand the operation of the various features including the SRAM, programmable clock, and I/O control.*

DESIGN FILES

The IP-EP2 Series Engineering Design Kit (EDK) includes a variety of files to assist the user in their development of the IP-EP2 Series Module. A summary of the various components of the EDK is given below.

Quartus II version 12.1 with Service Pack 1 (or later) is required.

The Quartus II project Ninek528 contains all of the files and settings necessary to implement the example design as described in the IP-EP2 Series User's Manual. The primary design files are listed below.

- **NineK528.vhd:** Top-Level Acromag provided VHDL (hardware design language) source file. Supports IP bus interface to ID, INT, and IO space.
- **Clkgene.vhdl:** Acromag provided VHDL source file. Supports the programming of the Cypress Clock IC.
- **DIG_IO_8.vhd:** Acromag provided VHDL source file. Supports 8 channels of digital change of state (COS) interrupts.
- **Ninek528.qsf:** Quartus II assignments file. This ASCII file contains all required user assignments included FPGA pin assignments and device options.
- **Ninek528.pof:** Altera specific configuration file. This file is generated by the Quartus II software and is used to directly program the FPGA via JTAG.
- **Ninek528.jic:** Altera specific configuration file. This file is generated by the Quartus II software and is used to program the FLASH device via JTAG.
- **Ninek528.hex:** Hexadecimal (Intel-Format) configuration file. The Hex file is an ASCII file in the Intel Hex format. This file is generated by the Quartus II software and is used to direct program the FPGA over the IP bus.
- **Ninek528.qpf:** Altera Quartus II specific master project file. Use to file to open the Quartus II example design project provided on the CD.
- **Ninek528.sdc:** timing constraints file (Synopsis Design Constraints)

In addition to the VHDL design files, the IP-EP2 Series Engineering Design Kit includes a schematic, parts list, parts location drawing, manuals, and other utility programs.

- **4502063a.pdf:** IP-EP2 Series Schematic and Part Location Drawing
- **IPEP2_797a.pdf:** IP-EP2 Series User's Manual
- **IPEP2_Programming_Guide.pdf:** IP-EP2 Series Engineering Design Kit Programming Guide.
- **IPEP201.pdf:** Part list for IP-EP201(E) model.
- **IPEP202.pdf:** Part list for IP-EP202(E) model.
- **IPEP203.pdf:** Part list for IP-EP203(E) model.
- **IPEP204.pdf:** Part list for IP-EP204(E) model.
- **HFileGenerator.exe:** This program generates 'C' style .h output file from an Intel.hex input file such as the NineK528.hex file. The .h file can be used to "compile in" the Altera configuration data into your own C program. This program can also be modified to allow programming of the IP-EP2 Series module over the IP bus. However you will need the base address of the IP-EP2 module in your system.
- **HFileGenerator.c:** Source C file for HfileGenerator.exe
- **BitCalc2k1.exe:** The BitCalc2k1.exe file is an executable program which provides the register values needed to program the clock generator chip. By entering the desired frequency, and selecting the IP clock speed (8MHz or 32MHz), this program will compute the correct values to write to the Clock control Registers.
- **IPEP2_Assignments.xls:** Summary of Quartus II assignments in excel format. The pin assignments can be copied directly into Quartus II.

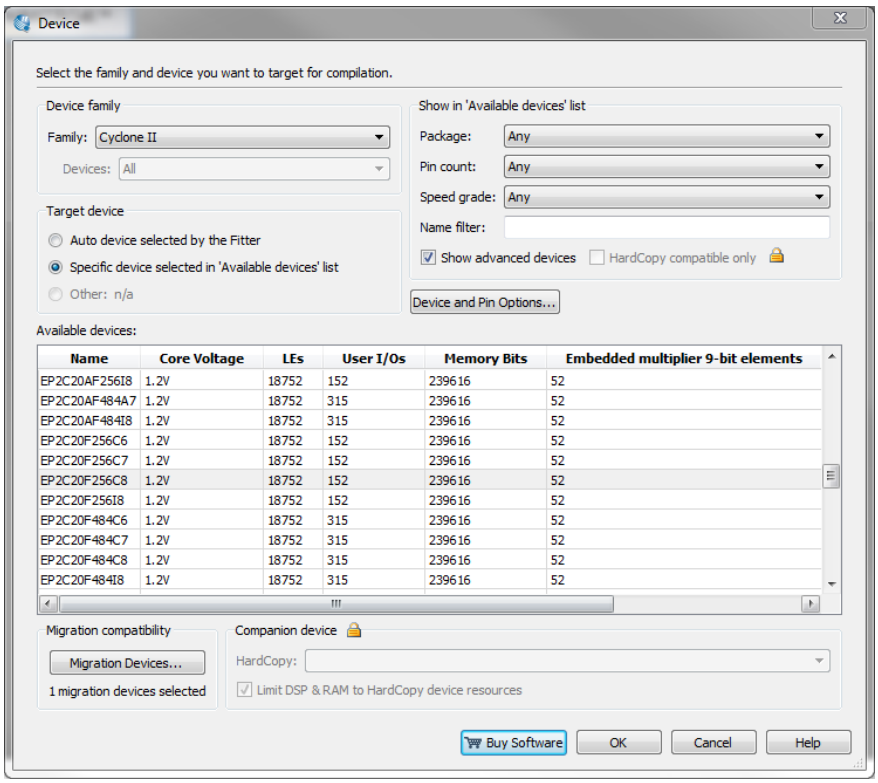
LOADING THE PROJECT

The example design project is provided in its entirety on the IP-EP2 Series Engineering Design Kit CD. To load the example design follow the steps detailed below.

1. Copy the "Quartus Project" folder located on the CD that accompanied the IP-EP2 EDK to the local hard drive.
2. Start the Quartus II Program. **WARNING Quartus II version 12.1 with Service Pack 1 (or later) is required to open this project.** The latest version of Quartus II is available, at no cost, for download at www.altera.com.
3. From Quartus II select *File->Open Project*. Then in the dialog box select the file ninek528.qpf from the folder. Click Open to complete the procedure.

PROJECT SETTINGS

Upon loading of the example design all project settings and assignments are present. If you have loaded the example design, you do not have to perform the following procedure and may skip to the *Required VHDL* section. However to familiarize yourself with the Quartus II software as well as the settings required for this board, it is recommended that you read this procedure and confirm that all settings are correct. A summary of all the project settings and assignments is available in the Appendix of this manual. The following procedure was written for Quartus II version 12.1 with Service Pack 1. *Note that the location of the settings and assignments may vary with newer versions of Quartus II.* Refer to Quartus II Help if you are unable to find a specific project setting.


Device
Select the family and device you want to target for compilation.
Device family
Family: Cyclone II
Devices: All
Target device
Auto device selected by the Fitter
Specific device selected in 'Available devices' list
Other: n/a
Show in 'Available devices' list
Package: Any
Pin count: Any
Speed grade: Any
Name filter:
Show advanced devices
HardCopy compatible only
Device and Pin Options...
Available devices:
Name | Core Voltage | LEs | User I/Os | Memory Bits | Embedded multiplier 9-bit elements
EP2C20AF256I8 | 1.2V | 18752 | 152 | 239616 | 52
EP2C20AF484A7 | 1.2V | 18752 | 315 | 239616 | 52
EP2C20AF484I8 | 1.2V | 18752 | 315 | 239616 | 52
EP2C20F256C6 | 1.2V | 18752 | 152 | 239616 | 52
EP2C20F256C7 | 1.2V | 18752 | 152 | 239616 | 52
EP2C20F256C8 | 1.2V | 18752 | 152 | 239616 | 52
EP2C20F256I8 | 1.2V | 18752 | 152 | 239616 | 52
EP2C20F484C6 | 1.2V | 18752 | 315 | 239616 | 52
EP2C20F484C7 | 1.2V | 18752 | 315 | 239616 | 52
EP2C20F484C8 | 1.2V | 18752 | 315 | 239616 | 52
EP2C20F484I8 | 1.2V | 18752 | 315 | 239616 | 52
Migration compatibility
Migration Devices...
1 migration devices selected
Companion device
HardCopy:
Limit DSP & RAM to HardCopy device resources
Buy Software
OK
Cancel
Help

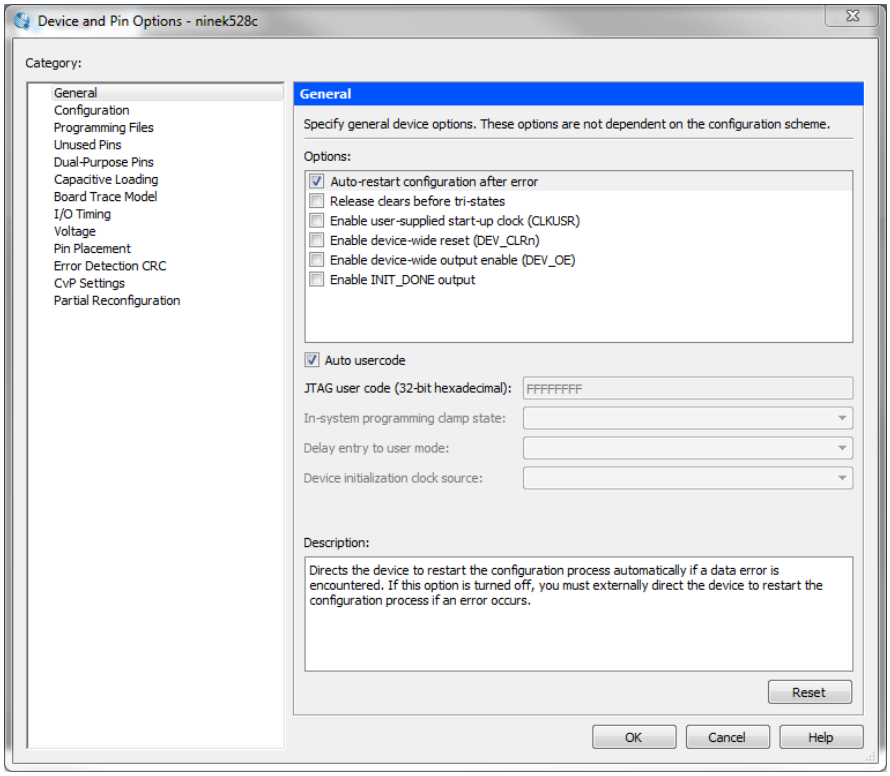
Device Settings

1. Access the Device Settings by selecting *Assignments ->Device*.

2. In the Device window select the Family as “Cyclone II”.

3. Select the “Specific device selected in ‘Available device’ list” radio button.

4. In the Available devices list select EP2C20F256C8. *EP2C20F256I8 can be selected for Industrial Temperature Rage modules.*



Device & Pin Options

5. From the Device page click the “Device & Pin Options...” button.

6. From the General Category check the following options: “Auto-Restart configuration after error”, and “Auto usercode.” All other options should NOT be selected.

7. The user may change the JTAG user code if desired. The default setting is Auto usercode.

8. Select the Configuration category.

Device Settings

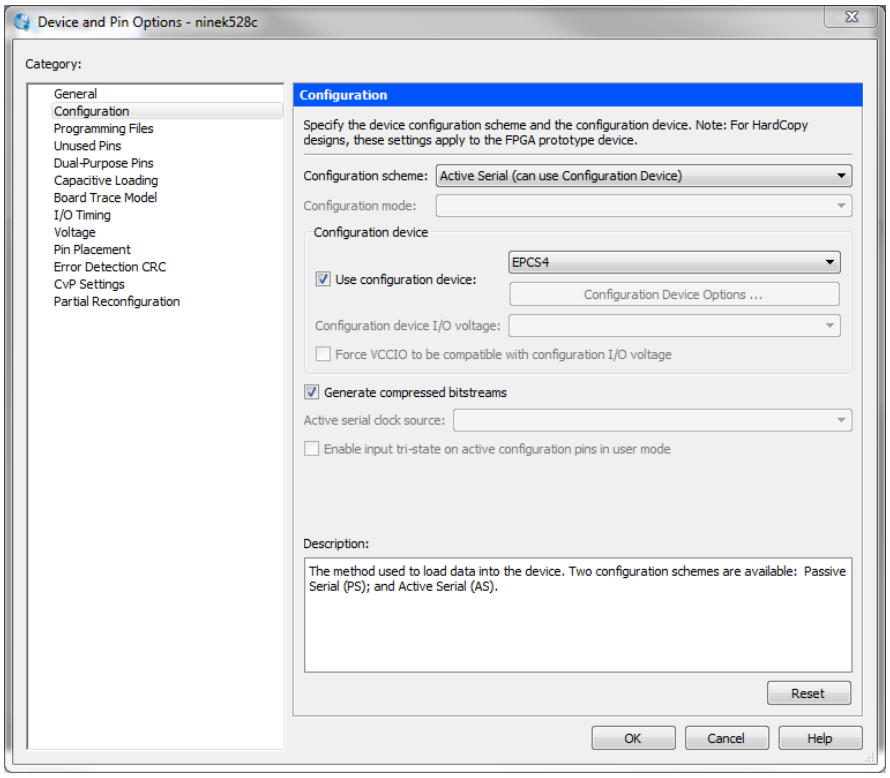
9. In the Configuration category select “Active Serial” as the configuration scheme.

10. Then check the “Use configuration device” box and select the “EPCS4 from the dropdown menu.

Note that either configuration scheme can be used with this setting. The Quartus II software selection simply reserves the appropriate pins.

11. The “Generate compressed bitstreams” is a user selectable option. The default option is on.

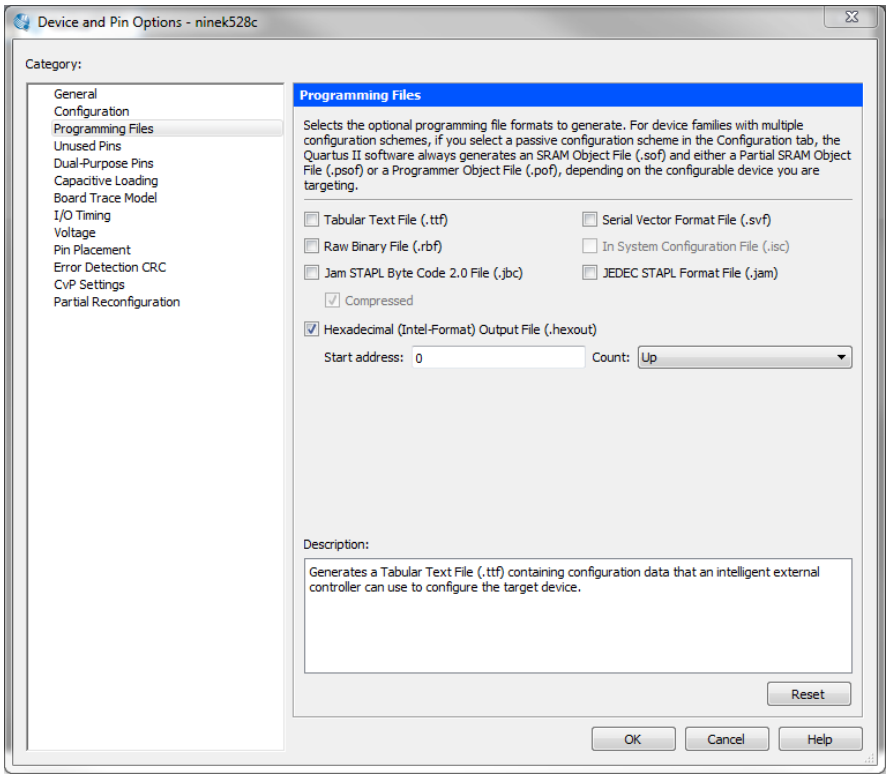
12. Select the Programming Files category

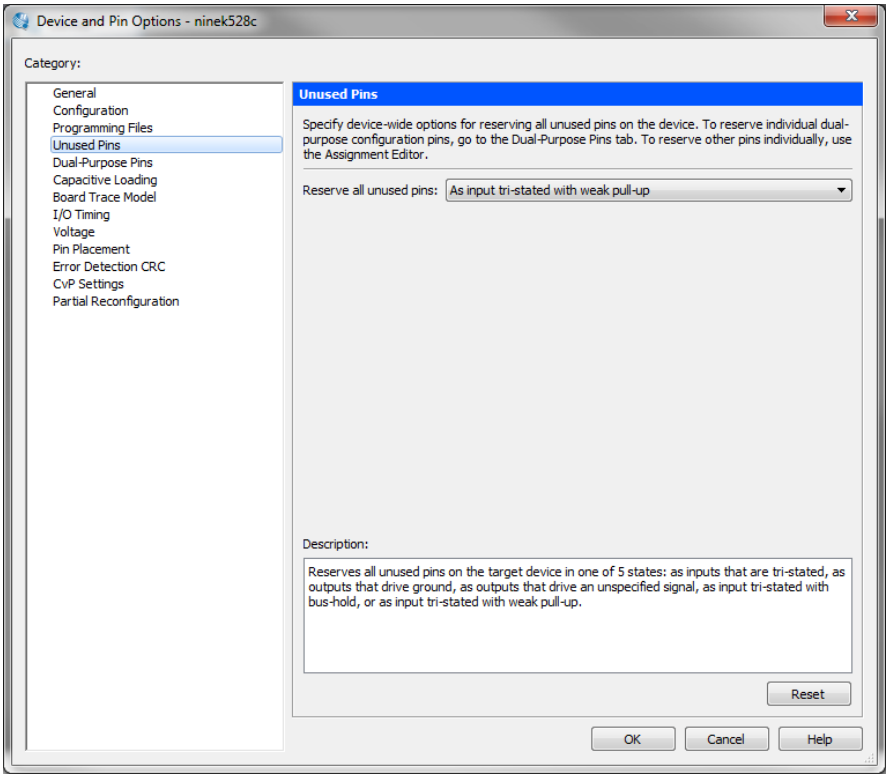


13. In the Programming Files category, check “Hexadecimal Output File.” Then set the Start address to 0 and the count as “up”.

The Intel Format Hexadecimal file is used when programming the FPGA over the IP bus. The FPGA direct JTAG programming file is always generated during compilation. The indirect FLASH programming file must be created separately. Refer to the FLASH programming portion of this manual for further details.

14. Select the Unused Pins category.

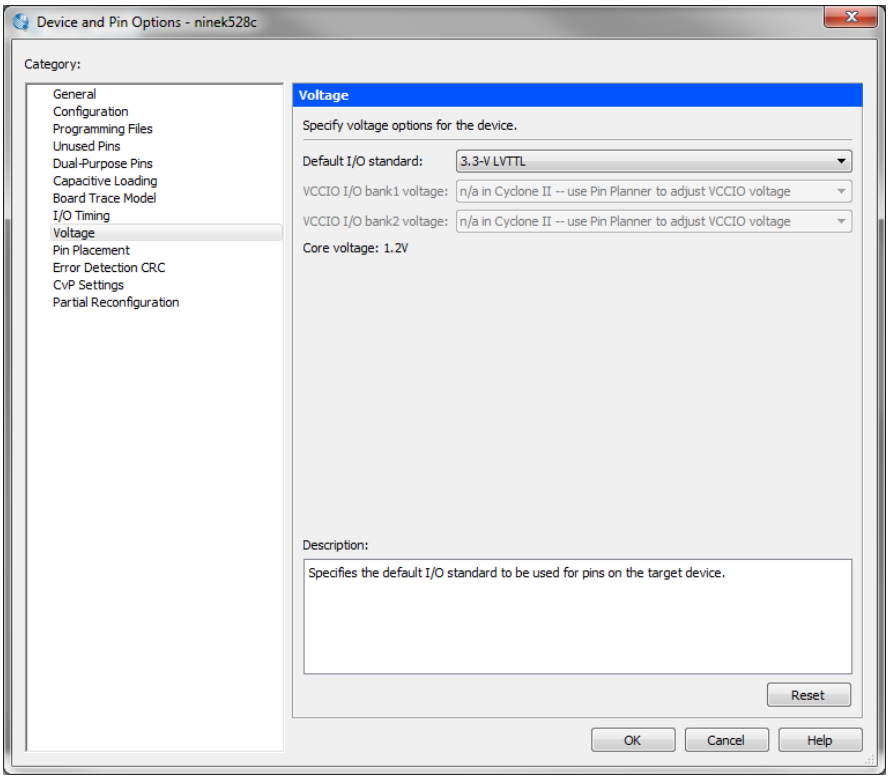

Device and Pin Options - ninek528c
Category:
General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Capacitive Loading
Board Trace Model
I/O Timing
Voltage
Pin Placement
Error Detection CRC
CvP Settings
Partial Reconfiguration
Programming Files
Selects the optional programming file formats to generate. For device families with multiple configuration schemes, if you select a passive configuration scheme in the Configuration tab, the Quartus II software always generates an SRAM Object File (.sof) and either a Partial SRAM Object File (.psof) or a Programmer Object File (.pof), depending on the configurable device you are targeting.
Tabular Text File (.ttf)
Serial Vector Format File (.svf)
Raw Binary File (.rbf)
In System Configuration File (.isc)
Jam STAPL Byte Code 2.0 File (.jbc)
JEDEC STAPL Format File (.jam)
Compressed
Hexadecimal (Intel-Format) Output File (.hexout)
Start address: 0
Count: Up
Description:
Generates a Tabular Text File (.ttf) containing configuration data that an intelligent external controller can use to configure the target device.
Reset
OK
Cancel
Help


Device and Pin Options - ninek528c
Category:
General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Capacitive Loading
Board Trace Model
I/O Timing
Voltage
Pin Placement
Error Detection CRC
CvP Settings
Partial Reconfiguration
Unused Pins
Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve other pins individually, use the Assignment Editor.
Reserve all unused pins: As input tri-stated with weak pull-up
Description:
Reserves all unused pins on the target device in one of 5 states: as inputs that are tri-stated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri-stated with bus-hold, or as input tri-stated with weak pull-up.
Reset
OK
Cancel
Help

Device Settings

15. In the Unused Pins category, reserve all unused pins “As inputs tri-stated with weak pull-up”.

WARNING: Due to the dual configuration nature of the IP-EP2 Series module, this option must be set correctly. Failure to do so may cause contention on the FPGA programming pins.



16. Select the Voltage category and set the default Voltage level to 3.3-V LVTTL in the pull-down menu.

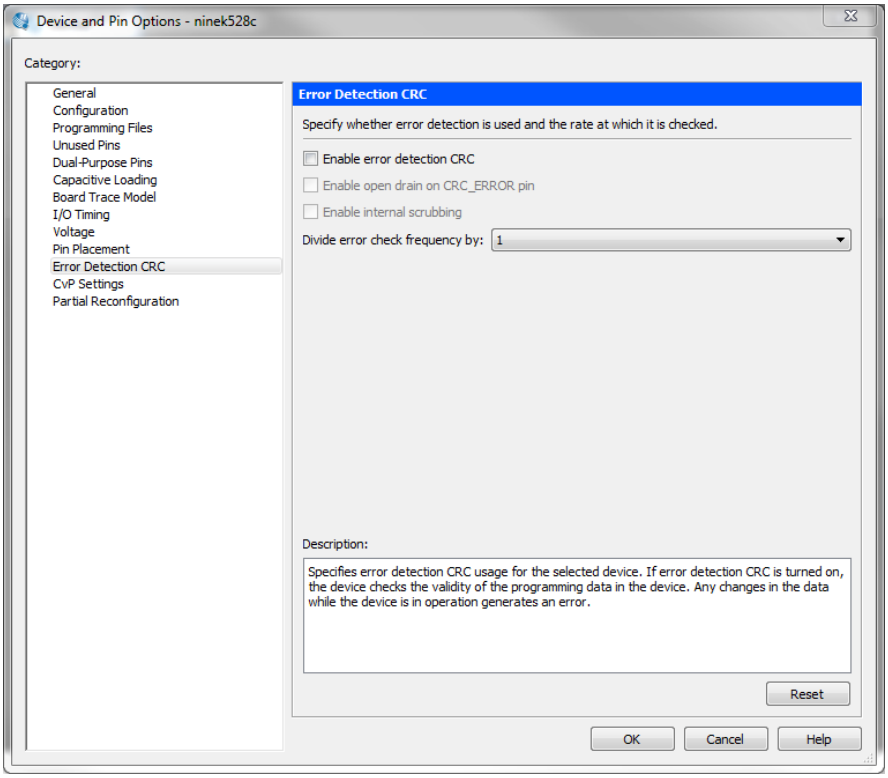
Device Settings

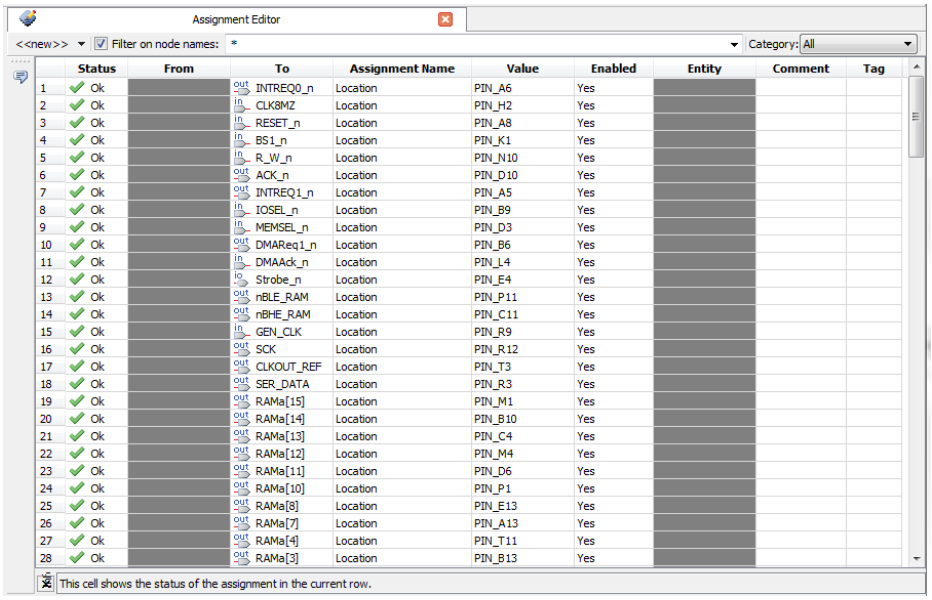
17. Select the Error Detection category and disable (uncheck) error correction. (User may optionally enable.)

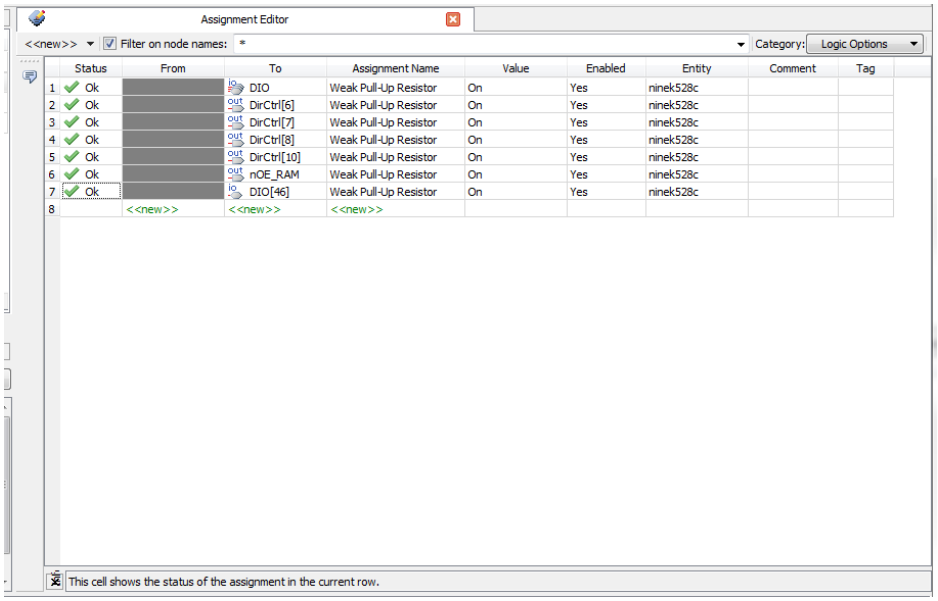
The remaining categories have no impact on any IP-EP2 module.

18. Click OK to close Device and Pin Options.

19. Click OK the close the Device page.


Device and Pin Options - ninek528c
Category:
General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Capacitive Loading
Board Trace Model
I/O Timing
Voltage
Pin Placement
Error Detection CRC
CvP Settings
Partial Reconfiguration
Error Detection CRC
Specify whether error detection is used and the rate at which it is checked.
Enable error detection CRC
Enable open drain on CRC_ERROR pin
Enable internal scrubbing
Divide error check frequency by: 1
Description:
Specifies error detection CRC usage for the selected device. If error detection CRC is turned on, the device checks the validity of the programming data in the device. Any changes in the data while the device is in operation generates an error.
Reset
OK
Cancel
Help


Assignment Editor
<<new>>
Filter on node names: *
Category: All
Status From To Assignment Name Value Enabled Entity Comment Tag
1 Ok INTREQ0_n Location PIN_A6 Yes
2 Ok CLK8MZ Location PIN_H2 Yes
3 Ok RESET_n Location PIN_A8 Yes
4 Ok BS1_n Location PIN_K1 Yes
5 Ok R_W_n Location PIN_N10 Yes
6 Ok ACK_n Location PIN_D10 Yes
7 Ok INTREQ1_n Location PIN_A5 Yes
8 Ok IOSEL_n Location PIN_B9 Yes
9 Ok MEMSEL_n Location PIN_D3 Yes
10 Ok DMAReq1_n Location PIN_B6 Yes
11 Ok DMAAck_n Location PIN_L4 Yes
12 Ok Strobe_n Location PIN_E4 Yes
13 Ok nBLE_RAM Location PIN_P11 Yes
14 Ok nBHE_RAM Location PIN_C11 Yes
15 Ok GEN_CLK Location PIN_R9 Yes
16 Ok SCK Location PIN_R12 Yes
17 Ok CLKOUT_REF Location PIN_T3 Yes
18 Ok SER_DATA Location PIN_R3 Yes
19 Ok RAMa[15] Location PIN_M1 Yes
20 Ok RAMa[14] Location PIN_B10 Yes
21 Ok RAMa[13] Location PIN_C4 Yes
22 Ok RAMa[12] Location PIN_M4 Yes
23 Ok RAMa[11] Location PIN_D6 Yes
24 Ok RAMa[10] Location PIN_P1 Yes
25 Ok RAMa[8] Location PIN_E13 Yes
26 Ok RAMa[7] Location PIN_A13 Yes
27 Ok RAMa[4] Location PIN_T11 Yes
28 Ok RAMa[3] Location PIN_B13 Yes
This cell shows the status of the assignment in the current row.


Assignment Editor
<<new>>
Filter on node names: *
Category: Logic Options
Status From To Assignment Name Value Enabled Entity Comment Tag
1 Ok DIO Weak Pull-Up Resistor On Yes ninek528c
2 Ok DirCtrl[6] Weak Pull-Up Resistor On Yes ninek528c
3 Ok DirCtrl[7] Weak Pull-Up Resistor On Yes ninek528c
4 Ok DirCtrl[8] Weak Pull-Up Resistor On Yes ninek528c
5 Ok DirCtrl[10] Weak Pull-Up Resistor On Yes ninek528c
6 Ok nOE_RAM Weak Pull-Up Resistor On Yes ninek528c
7 Ok DIO[46] Weak Pull-Up Resistor On Yes ninek528c
8 <<new>> <<new>> <<new>>
This cell shows the status of the assignment in the current row.

IP-EP2 ASSIGNMENTS

The example design contains all of the assignments required for the proper operation of the module. Instructions for setting the assignments are below.

20. Select *Assignments-> Assignments Editor*.

21. In the Category Window click on “All”.

22. In the pin spreadsheet, place the signal name in the “To” column and the pin alphanumeric id in the Value column.

23. Enter all pin names and locations as given in the Pin Assignments Table in the *Appendix*.

24. In the Category Windows click on “Logic Options”.

25. In the Spreadsheet, double click on <<new>> in the To column. Then enter *DIO*. Then double click on the Assignment Name column and select *Weak Pull Up Resistor* from the pull-down menu. Confirm that the Value column is set to *On* and the Enabled column is set to *Yes*.

26. Repeat Step 25 with the following names in the “To” column: DirCtrl[6], DirCtrl[7], DirCtrl[8], DirCtrl[9], DirCtrl[10], and DirCtrl[11]. These signals require pull-up resistors.

All assignments are summarized in the Appendix.

REQUIRED VHDL

This section highlights the functionality of some of the VHDL in the Acromag example.

After the Project settings and assignments have been set or verified, the user should become familiar with the IP-EP2 Series board prior to modifying the VHDL. They should understand the IP Bus and Cypress Programmable clock interface, the asynchronous interface with the SRAM, and learn how the I/O are controlled. Acromag recommends that you do not directly modify the interface with either the IP bus or the Cypress Clock. If it is necessary to modify either interface be sure to completely understand the requirements as defined in the IP specification or the appropriate data sheet. Failure to do so may cause the board to stop responding to IP bus requests.

In addition to these interfaces, several control signals to the CPLD must be maintained by the FPGA. These signals, one to enable configuration mode, and another to assist with IP bus control must be included in any program targeting the FPGA.

EnableCPLD Configuration Control Signal

WARNING: MODIFYING THE VHDL SOURCE CODE FROM THE EXAMPLE DESIGN COULD RESULT IN BOARD FAILURE! BE SURE TO SIMULATE AND UNDERSTAND ANY MODIFICATION PRIOR TO ITS IMPLEMENTATION.

There are two main modes of operation on the IP-EP2 Series module: configuration mode and user mode. The IP-EP2 Series powers up in configuration mode and remains in that mode until the Altera FPGA is successfully configured. Once the Altera FPGA is successfully configured, control is automatically transferred to user mode and the Altera FPGA has control of the IP bus interface. In order to implement this transition, the following requirements must be respected by the Altera FPGA.

1. Pin L3 of the Altera FPGA is reserved as an EnableCPLD control. When Pin L3 is driven low the IP-EP2 module is in user mode and the Altera FPGA has control of the IP bus interface. When Pin L3 (EnableCPLD) is driven high the IP-EP2 module is in configuration mode.
2. The EnableCPLD signal (Pin L3) should be driven by Altera FPGA logic similar to that shown in the following VHDL process. Notice that after the Altera FPGA is configured the EnableCPLD signal is driven to a logic low by the configured Altera FPGA. A logic low holds the IP-EP2 Series module in user mode.
3. The EnableCPLD signal (L3) can be driven to a logic high via an IP bus write cycle to base address + 0 hex with Data line 0 set high. Setting EnableCPLD high returns the IP-EP2 module to configuration mode. Note that this procedure is only required when programming the FPGA over the IP bus. The JTAG interface will automatically disable the FPGA and hand control over to the CPLD.

```
process (CLK8MZ, RESET)
begin
        if (RESET = '1') then
                EnableCPLD_Reg <= '0';
        elsif (CLK8MZ'event and CLK8MZ = '1') then
                if (WR_Ctrl_L = '1') then
                        EnableCPLD_Reg <= DLOW(0);
                else
                        EnableCPLD_Reg <= EnableCPLD_Reg;
                end if;
        end if;
end process;
```

REQUIRED VHDL

```
process (CLK8MZ, RESET)
 begin
        if (CLK8MZ'event and CLK8MZ = '1') then
              EnableCPLD <= (EnableCPLD_Reg and not ACK) or
                            (EnableCPLD_REG and EnableCPLD);
        end if;

 end process;
```

Note: Using the above VHDL will delay changing the EnableCPLD signal until after the IP-EP2 module has acknowledged the write to the control register. This prevents the IP bus from locking due to an unacknowledged write cycle.

4. After the Altera FPGA has returned control to the configuration mode, the CPLD will take over control of the IP bus. Once disabled the CPLD will not pass any IP bus signals from the FPGA to the carrier. However the I/O and interrupts will still function as last programmed. Since the CPLD will not be able to handle an interrupt request, it is recommended that all interrupts be disabled prior to re-entering configuration mode. Once Configuration Mode is enabled in this manner the only way to return back to user mode is to reconfigure the board or by issuing an IP module Reset from the carrier board.

IPRead_Enable Data Bus Direction Control Signals

WARNING: FAILURE TO CONTROL THESE SIGNALS PROPERLY MAY RESULT IN A BUS CONFLICT!

The Altera FPGA requires buffers between itself and the IP bus. This is due to the fact that the FPGA is not 5V tolerant and the IP bus specification is based upon 5V signaling levels. Buffering is not an issue for any unidirectional signal. However, the IP data bus is bi-directional and as such requires a direction control signal at the buffer. This is accomplished through the use of two signals on the FPGA, IPRead_En_Low and IPRead_En_High. The IPRead_En_Low signal is the direction control for the lower eight data bits D0 to D7. The IPRead_En_High signal is the direction control for the upper eight data bits D8 to D15. The data flow direction is indicated by the table below. In most cases these signals will be identical.

Signal	Value	Data Flow
IPRead_En_Low/ IPRead_En_High	0	IP BUS-> FGPA (Write operation) Default
	1	FPGA -> IP BUS (Read operation)

The IP BUS->FPGA data flow should be the default since this will avoid potential data bus conflicts. The only time the data flow should from the FPGA to the IP bus is during either I/O Space, Identification (ID) Space, Interrupt Space, or Memory Space read cycles. This is done in the software by taking the logical “Or” of each of these cycles. Refer to the example design vhdl code to observe how each read cycle is identified and then passed to the two direction control variables.

Compilation

At this point, the user can compile the example design. If you are already familiar with the capabilities and specifications of the IP-EP2 Series modules then modify the VHDL as desired. The design can be compiled by selecting *Start Compilation* from the *Processing* menu. After the process has finished correct any errors and review the warning statements. Review the compilation report that will appear on the screen. This is also an ideal time to simulate the FPGA using the Quartus II built-in simulator. The procedure for simulation is beyond the scope of this manual, but it is well documented in the Quartus II help files. The next step is to program the board. Please refer to the appropriate method for detailed instructions on programming the board.

2.0 PROGRAMMING THE BOARD

The IP-EP2 Series module has three methods of configuration. The first is configuring the Altera FPGA directly over the IP Bus. This method uses the passive serial scheme to directly program the FPGA. Note that this technique requires the FPGA to be reprogrammed at power-up. The second method is to configure the part directly using the JTAG interface. The JTAG interface will automatically over-write any existing configuration and can be completed at any time using a standard Altera JTAG download cable such as the ByteBlaster 2®. This cable is NOT provided by Acromag. Once again all programming is lost at power-down using the direct JTAG configuration approach. Finally the IP-EP2 Series module contains a Flash Configuration Device (Altera EPCS4 or equivalent) that can be programmed indirectly through the JTAG interface using the Altera Serial Flash Loader. The Serial Flash Loader creates a logic bridge between the Cyclone II JTAG interface and the controls of the FLASH device. This bridge allows the user to program the Flash via the JTAG interface. The FLASH device cannot be programmed through the IP interface. This method is recommended for debugged designs since the Flash device programs the Altera FPGA at power-up. The programming procedures for each of the three methods are below.

FPGA CONFIGURATION OVER THE IP BUS

The Cyclone II FPGA can be programmed directly over the IP bus. To program the Cyclone II FPGA over the IP bus follow the procedure below.

1. Generate the Intel hexadecimal programming file. This file is generated automatically upon compilation in your Quartus II project directory if the correct option is selected in the Programming Files Tab under Device Options. To generate an Intel hexadecimal file manually in Quartus II, select Convert Programming Files under the File Menu. Then under programming file type select Hexadecimal (Intel-format). Set the output file name and directory as desired. Then add the .sof file from your project. If no .sof file exists, then the project has not yet compiled to completion. Once the .sof file has been added, click Generate. Close the window when finished. Note that the hexadecimal file must be accessible by the computer that contains the IP-EP2 Series module.
2. Power-down the computer with the IP-EP2 module and set the Configuration Jumper to “IP BUS” as shown in JTAG Interface/Jumper Location drawing located in the IP-EP2 Series User’s Manual. Failure to set this jumper correctly will cause programming to fail.
3. Power-up the system. Upon system power-up the IP-EP2 Series module is in configuration mode. If the Altera FPGA is currently configured and operational, configuration mode can be entered by driving pin L3 of the Altera FPGA to a logic high via the control register bit-0. Pin L3 is the Config_Enable signal which upon system power-up is held high by a pullup resistor.
4. You can verify that you are in configuration mode by reading ID space at base address + 0AH. The byte read will be 48H when in configuration mode and 49H when in user mode.
5. Configuration is started by setting bit-0 of the control register, at base address + 00H, to a logic high.

FPGA CONFIGURATION OVER THE IP BUS

6. This same register bit-0 must be read next. When read as a logic high software can proceed to the data transfer phase. A polling method should be used here since this bit will not be read high until at least 40µ seconds after the control bit is set high.
7. The status of the Altera FPGA during configuration can be monitored via the Status register at base address + 00H. Bit-1 monitors the Altera nStatus signal which must remain high during configuration. Bit-2 of the Status register reflects the Altera FPGA CONF_DONE signal. The CONF_DONE signal must remain at a logic low until configuration has completed.
8. Write program data from the Intel Hexadecimal file, one byte at a time, to the Configuration Data register at base address + 02H.
9. Upon successful configuration, control of the IP bus will automatically be switched to user mode and the Altera FPGA will have control of the IP bus interface. It is good practice to issue a software reset prior to operating the board.

Note that all configuration data will be lost when the board is powered down.

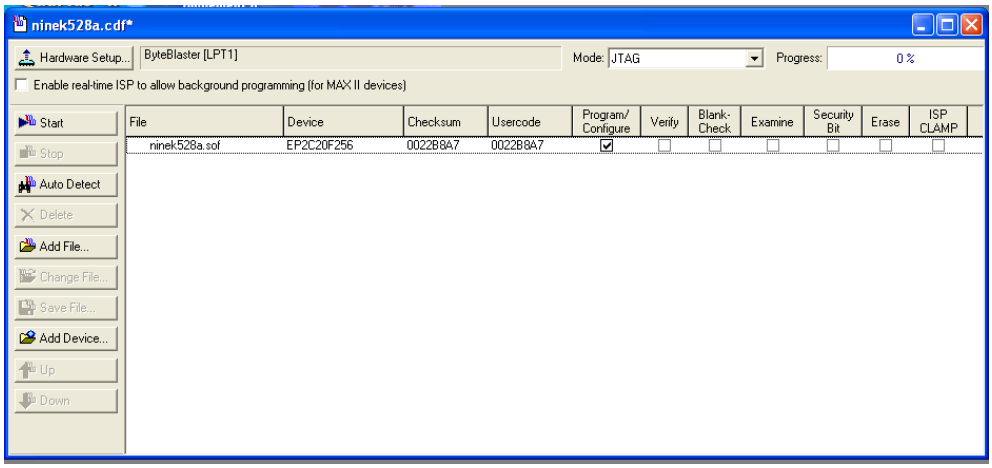
DIRECT FPGA CONFIGURATION VIA JTAG

The IP-EP2 Series Cyclone II FPGA can be configured using a standard JTAG interface. The JTAG interface can either program the FPGA directly or program the FLASH configuration memory. When programming the FPGA directly, the programming jumper may be in either position.

The following is the general procedure for direct programming of the Altera FPGA using the JTAG interface.

1. Generate the .sof programming file. This file is automatically generated by Quartus II upon successful compilation. The file is located in your Quartus II project directory.
2. Power-down the IP-EP2 module and connect the 10-pin Altera JTAG cable (not included) to the board. This cable is available from Altera.
3. Power-up the IP-EP2 module.
4. Start the Quartus II Programmer. The Programmer can be started by first starting the Quartus II software and then selecting Programmer under the Tools menu.
5. In the Programming Window, click Hardware Setup. Under the Currently Selected Hardware pull-down menu, select the device that connects to the IP-EP2 board (i.e. ByteBlaster II). Click close to return to the Programming Window.
6. From the Mode pull-down window, select JTAG.
7. In the left pane, click the Add File button. Then select the *.sof file generated in step 1. Click Open. Now the programming file and the Cyclone II device should be listed in the window.
8. Check the Program/Configure check box.
9. Then click on the Start button to download the file to the FPGA via JTAG.
10. Upon successful configuration the board will be in User mode with the Altera FPGA in control of the IP bus interface. It is good practice to issue a software reset prior to operating the board.

Note that all configuration data will be lost at power down.



DIRECT FPGA CONFIGURATION VIA JTAG

Quartus II Programmer Example for direct FPGA JTAG Programming.

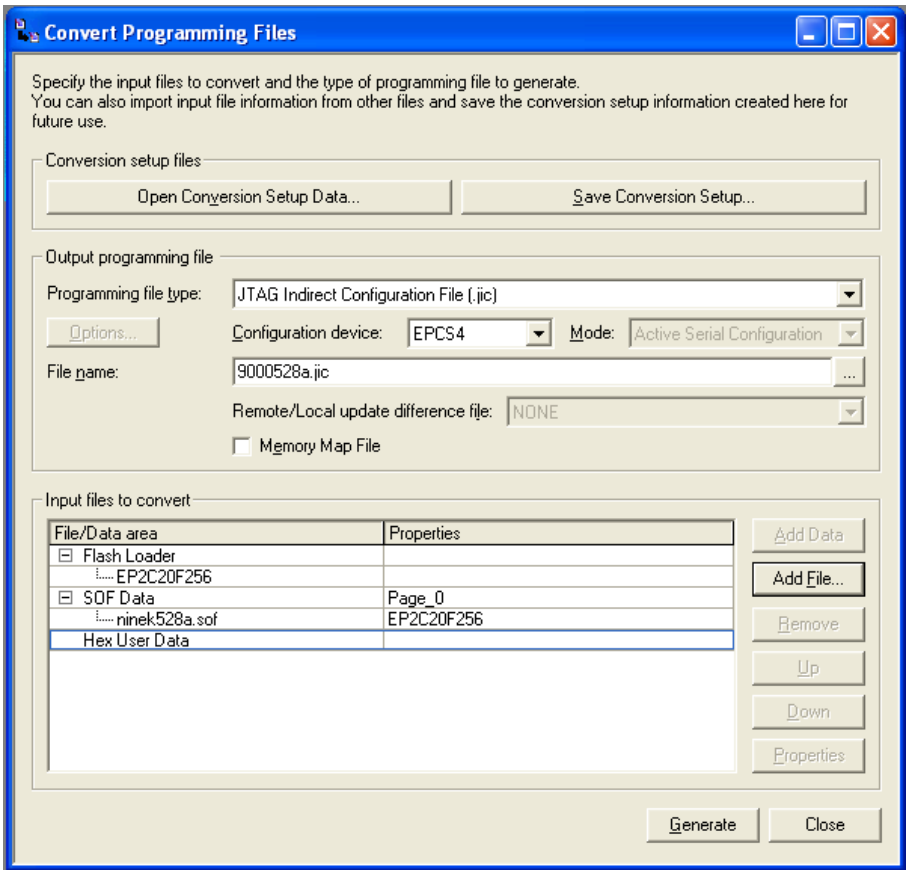
FLASH CONFIGURATION VIA JTAG

The IP-EP2 Series module can also implement configuration using a standard JTAG interface. The JTAG interface can either program the FPGA directly or program the FLASH configuration memory. Note that the FPGA will require reprogramming after power down.

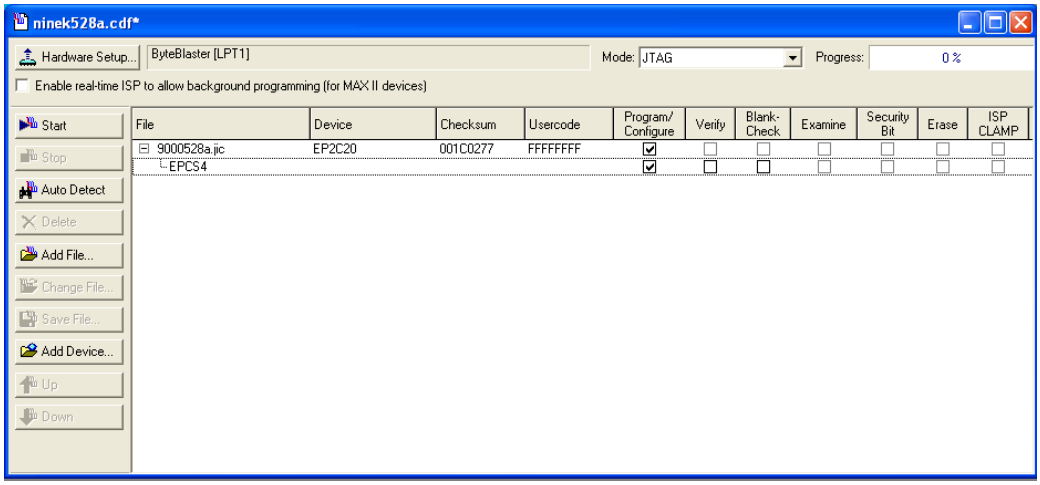
1. Power-down the IP-EP2 module and connect the 10-pin Altera JTAG cable (not included) to the board. This cable is available from Altera.
2. Set the Configuration Jumper to “FLASH” as shown in JTAG Interface/Jumper Location drawing located in the IP-EP2 Series User’s Manual. Failure to set this jumper correctly will cause programming to fail.
3. Power-up the IP-EP2 module and start the Quartus II software.
4. Generate the .sof programming file. This file is automatically generated by Quartus II upon successful compilation. The file is located in your Quartus II project directory.
5. Select *Convert Programming Files* from the *File* menu.
6. In the *Convert Programming Files* dialog box, select JTAG Indirect Configuration File (.jic) from the *Programming file type* pull-down menu.
7. In the *Configuration Device* pull-down menu select EPCS4.
8. In the *File name* field, set the output file name and directory.
9. Click on “SOF Data” in the Input Files to convert section.
10. Click Add File and select the sof file generated in step 1. Click OK
11. Highlight FlashLoader and click Add Device.
12. Select the Cyclone II EP2C20 device. Click OK.
13. Click the Generate process to create the JIC file.

FLASH CONFIGURATION VIA JTAG

Convert Programming Files Dialog for example JIC creation.



14. Start the Quartus II Programmer. The Programmer can be started by first starting the Quartus II software and then selecting Programmer under the Tools menu.
15. In the Programming Window, click Hardware Setup. Under the Currently Selected Hardware pull-down menu, select the device that connects to the IP-EP2 Series board (i.e. ByteBlaster II). Click close to return to the Programming Window.
16. From the Mode pull-down window, select JTAG.
17. In the left pane, click the Add File button. Then select the *.jic file generated in step 5. Click Open. Now the programming file and the Cyclone II device should be listed in the window.
18. Check both boxes under the Program/Configure column.
19. Then click on the Start button to download the file to the FLASH via JTAG.



FLASH CONFIGURATION VIA JTAG

Quartus II Programmer Example for Flash configuration.

20. Once complete the IP-EP2 module will still be in configuration mode. To trigger a configuration cycle to load the program from Flash, either write a “1” to bit 0 of the Configuration Control/Status Register (Base Addr + 0x0) or power down and then power the board back up.

3.0 HARDWARE PROGRAM DISABLE

WARNING: REMOVING HARDWARE FROM BOARD MAY VOID ACROMAG WARRANTY!

WARNING: PROCEDURES REQUIRE THE REMOVAL OF SMALL SURFACE MOUNT COMPONENTS!

WARNING: ONCE PART IS REMOVED, THE BOARD CAN NOT RE REPROGRAMMED WITHOUT A REPLACEMENT!

This section provides instruction in how to disable all device programming. *This section is for advanced users only.* These procedures should be used *only* when it is critical to remove the ability to reprogram the IP-EP2 module. Note that to remove all programming abilities the user program must reside in FLASH. These procedures require the use of the part location drawing provided on the Engineering Design Kit CD.

WARNING: These procedures require the removal of small surface mount components! Removing these parts may void the Acromag warranty! Furthermore, once these parts are removed, the board cannot be reprogrammed without replacing these parts.

Disable Programming over the IP Bus

Remove 0 Ohm Resistors R32 and R33. This will disconnect the programming data and clock lines from the CPLD to the FPGA. Once removed programming over the IP bus will no longer function. Do not attempt programming after these resistors have been removed. Note that there is no method to verify resistor removal through software.

Disable JTAG Programming

Remove 0 Ohm Resistors R34 and R37. This will disconnect the data and clock JTAG programming lines. Once removed the JTAG connection will no longer function. All JTAG operations will be disabled. Note that there is no method to verify resistor removal through software.

Replacing the Configuration Jumper

The configuration jumper can be replaced with a 0 Ohm resistor. This would fix the programming methodology to a single method. Please contact Acromag for more information on ordering boards with a fixed programming method.

4.0 TROUBLESHOOTING

FREQUENTLY ASKED QUESTIONS

This section contains some of the frequently asked questions regarding the IP-EP2 series modules. This is by no means an exhaustive list. Users can also consult with the IP-EP2 Series User's Manual, Quartus II Help, and the part Data Sheets for further information.

What do I need to implement an IP-EP2 Series module?

To program the IP-EP2 Series module, you will need Acromag's Engineering Design Kit (IP-EP2-EDK). For easier integration with your operating system, we also recommend our OS software support packages for Linux, VxWorks, and Windows.

The Engineering Design Kit includes schematics for the boards, example VHDL code, and example software for downloading the Hex code (converted from VHDL code) to the FPGA on the IP-EP2 module. It does not contain the VHDL design software. The most commonly used design tool used for this purpose is the QUARTUS II software, which is downloadable at no cost from Altera. This free software includes a VHDL compiler, timing analysis tools, and more. Additionally, the EDK does not provide an Altera JTAG download cable. This cable is available for purchase from Altera. Contact Altera for guidance on which software package or download cable would be appropriate for your use with the Cyclone II FPGA.

Helpful Tip: *Users should be familiar with the Quartus II software prior to modifying the IP-EP2 firmware. If not, run the Quartus II tutorial available from the Help Menu.*

Where can I find information on the Cyclone II FPGA?

Documentation on the Cyclone II FPGA is available from Altera's website at www.altera.com/literature/lit-cyc2.jsp.

Where can I find information on the SRAM or Clock Generator IC?

Documentation of the SRAM IC is available from IDT's website at www.idt.com. Then search for the part 71V016SA.

Documentation of the Cypress Clock Generator is available from the Cypress website at www.cypress.com. Then search for the part CY22150.

Where can I find information on the IP Bus Interface?

The IP bus specification ANSI/VITA 4-1995 is available for purchase from www.vita.com.

Why does the IP-EP2 module require me to implement wait states?

The Cyclone II FPGA has a buffer between itself and the IP bus. This buffer allows for 5V signaling on the IP carrier. Unfortunately, the buffer adds an additional 10ns maximum of propagation delay for all IP signals. The propagation delay may not allow bus signals to settle during 32MHz operations. As such 1 wait state is required for all IP module read/write operations to take into account the additional delay of the buffer.

Helpful Tip: *If the board does work correctly, try downloading the example program. If this works, then simulate your firmware to try to find the problem.*

FREQUENTLY ASKED QUESTIONS

How does the Configuration Jumper setting work?

The configuration jumper controls the voltage of the Cyclone II MSEL0 pin. This jumper ties the pin to either 3.3V or ground. The pin controls where the FPGA receives its configuration data. If the pin is high, the FPGA is set in Passive Serial (PS) mode and the configuration data is passed from the IP bus. If the pin is set low, the FPGA is in Active Serial (AS) mode and the FPGA is configured directly from FLASH memory.

How should the Configuration Jumper be set when I am using the JTAG cable?

Helpful Tip: *The Quartus II MegaWizard Plug-in Manager can create functional blocks for many common components such as FIFO's or internal RAM.*

When programming the FLASH memory via JTAG, the jumper must be set in the FLASH position. If using the JTAG cable to either direct program the FPGA or when using the SignalTap II debugger, the jumper can be in either position.

What can I do if I cannot communicate with the IP module after I download a custom program?

If the IP module does not respond, then there is likely a problem with the VHDL controlling the IP interface. The first step should be to download the example program. Once the IP-EP2 module is reconfigured, test the IP interface with the sample program provided in the Acromag OS software support packages. If the example program interface functions, then you can start debugging your own code. Use the Quartus II functional simulator to emulate a sample IP bus cycle. Another option is to use SignalTap II, a JTAG debugger, though modification of the VHDL will be necessary.

How do I implement the Signal Tap II debugger.

Note to enable SignalTap II in the Quartus II Web Edition, the Altera "TalkBack" feature must be enabled. Refer to Altera documentation for more information on this program.

SignalTap II is a FPGA debugging tool that allows the user to debug the firmware under real operation conditions.. The debugger interfaces with the Cyclone II FPGA on the IP-EP2 modules via the JTAG connection. To use this feature an Altera JTAG download cable is required. Acromag does not directly support the SignalTap II debugger, though it can be integrated into our example design. The following procedure is a brief introduction on using SignalTap II and is provided for reference only.

The MegaWizard Plug-In Manager under the Tools menu can be used to create an instance of the SignalTap II debugger in the VHDL code. Within the Wizard, select the Cyclone II family, as well as the memory depth and the number and type of triggers. After running the wizard and integrating the newly creating component into the design, compile the program. Then set up a STP file for each instance of a SignalTap analyzer by the *Create Signal Tap II file from Design Instance* command under the File->Create/Update menu. Refer to the Quartus II Help files for more information on this procedure and using the SignalTap II Logic analyzer.

FREQUENTLY ASKED QUESTIONS

What configuration mode do I select in the Quartus II software?

Acromag recommends that you select the AS configuration mode regardless of the configuration method used. This option in the Quartus II software only reserve pins. The actual configuration method is defined at power-up via the configuration jumper on the IP-EP2 series board.

Can I program the flash memory over the IP bus?

Acromag only supports programming the flash device over the JTAG connection. However, Altera provides a Megafunction *altasmi_parallel* (Active Serial Memory Interface Parallel) to allow the Cyclone II FPGA to access the flash. This library component could be used to program the flash or to utilize the remaining space for non-volatile memory.

Why do I need to write 01H to the Configuration Control Register after JTAG programming the FLASH?

Programming the FLASH via JTAG requires that a "bridge" program be loaded into the FPGA that allows the JTAG signals to control the serial flash interface. Upon completion of programming the FLASH device, the FPGA is *not* reset and still contains the "bridge" program. As such the IP-EP2 board will be in configuration mode. Writing a 01H to the Configuration Control Register will then instruct the FPGA to reload its new program from FLASH.

Is the Strobe_n IP bus signal accessible on the IP-EP2 and, if available, how does it work?

Yes, the Strobe_n signal is available for use on all Revision B (or later) IP-EP20x models. Contact Acromag for information on determining your products revision level. The Strobe_n signal is routed from the FPGA to the CPLD and then to the IP bus. Due to the intermediary stop on the CPLD a direction control signal is required. This control signal is called Strobe_Dir in the example design and must be set properly. Logic low ('0') sets the Strobe_n signal as an input and logic high ('1') sets the Strobe_n signal as an output.

Does Acromag provide a test bench?

Acromag currently does not provide test benches for these models since few uses retain the exact functionality of the example design. We recommended that you simulate the design using the built in simulator within Quartus II. The ANSI/VITA 4 Industry Pack bus specification provides timing diagrams to use for the simulation.

Why are there over 100 warnings when I compile the example design?

The majority of these warnings are related to I/O pins that are defined as pins but not utilized or fixed at a certain logic level within the example design. Examples include the IP bus DMA control signals and strobe signals.

Can we utilize the Error Signal on the IP bus?

No. The Error signal is reserved for factory use and is pulled high during normal operation. Consider using interrupts to indicate error conditions.

APPENDIX

PIN ASSIGNMENTS

The pin assignments as well as a brief description and the corresponding name in the schematic and VHDL file are detailed in the Pin Assignments table. A similar table is also provided in Excel format on the IP-EP2 EDK CD.

Note that the pin location is preceded by "Pin_".

*Pin assignments are stored in the project *.qsf file.*

Note that schematic connection names preceded by ~ are active low signals

Field I/O signals are either a bi-directional I/O line to a transceiver or a direction control line.

TTL inputs on the EP201 are a one to one match to the DIO bus signals. The TTL direction is controlled in groups of eight via DirCtrl bits 6 to 11.

The even numbered DIO bus channels are mapped to the differential transceivers. This includes all differential channels on the EP202 and EP204. The differential direction is controlled in groups of four via DirCtrl bits 0 to 5.

The EP203 maps to the lower 24 TTL channels and the upper 12 differential channels.

The exact mapping of the direction control can be found in the IP-EP2 Series User's Manual.

Pin Assignments Table

VHDL Name	Pin #	Direction	Schematic Connection
FIELD I/O SIGNALS			
DIO[0]	PIN_R6	Bidir	DIG_D0
DIO[1]	PIN_G12	Bidir	DIG_D1
DIO[2]	PIN_T5	Bidir	DIG_D2
DIO[3]	PIN_M2	Bidir	DIG_D33
DIO[4]	PIN_T13	Bidir	DIG_D4
DIO[5]	PIN_K5	Bidir	DIG_D5
DIO[6]	PIN_R13	Bidir	DIG_D6
DIO[7]	PIN_P5	Bidir	DIG_D7
DIO[8]	PIN_C13	Bidir	DIG_D8
DIO[9]	PIN_G13	bidir	DIG_D9
DIO[10]	PIN_P13	Bidir	DIG_D10
DIO[11]	PIN_B12	Bidir	DIG_D11
DIO[12]	PIN_N15	Bidir	DIG_D12
DIO[13]	PIN_A12	Bidir	DIG_D13
DIO[14]	PIN_M15	Bidir	DIG_D14
DIO[15]	PIN_D16	Bidir	DIG_D15
DIO[16]	PIN_M14	Bidir	DIG_D16
DIO[17]	PIN_P15	Bidir	DIG_D17
DIO[18]	PIN_B11	Bidir	DIG_D18
DIO[19]	PIN_F14	Bidir	DIG_D19
DIO[20]	PIN_T4	Bidir	DIG_D20
DIO[21]	PIN_M16	Bidir	DIG_D21
DIO[22]	PIN_K15	Bidir	DIG_D22
DIO[23]	PIN_G15	Bidir	DIG_D23
DIO[24]	PIN_D14	Bidir	DIG_D24
DIO[25]	PIN_R7	Bidir	DIG_D25
DIO[26]	PIN_P12	Bidir	DIG_D26
DIO[27]	PIN_K13	Bidir	DIG_D27
DIO[28]	PIN_F16	Bidir	DIG_D28
DIO[29]	PIN_G16	Bidir	DIG_D29
DIO[30]	PIN_A14	Bidir	DIG_D30
DIO[31]	PIN_D9	Bidir	DIG_D31
DIO[32]	PIN_C16	Bidir	DIG_D32
DIO[33]	PIN_T14	Bidir	DIG_D33
DIO[34]	PIN_D15	Bidir	DIG_D34
DIO[35]	PIN_R8	Bidir	DIG_D35
DIO[36]	PIN_K16	Bidir	DIG_D36
DIO[37]	PIN_D13	Bidir	DIG_D37

VHDL Name	Pin #	Direction	Schematic Connection
DIO[38]	PIN_F13	Bidir	DIG_D38
DIO[39]	PIN_J12	Bidir	DIG_D39
DIO[40]	PIN_N16	Bidir	DIG_D40
DIO[41]	PIN_H12	Bidir	DIG_D41
FIELD I/O SIGNALS			
DIO[42]	PIN_L16	Bidir	DIG_D42
DIO[43]	PIN_P16	Bidir	DIG_D43
DIO[44]	PIN_D8	Bidir	DIG_D44
DIO[45]	PIN_N7	Bidir	DIG_D45
DIO[46]	PIN_F15	Bidir	DIG_D46
DIO[47]	PIN_B3	Bidir	DIG_D47
DirCtrl[0]	PIN_T12	Output	DIFF_DIR0
DirCtrl[1]	PIN_R14	Output	DIFF_DIR1
DirCtrl[2]	PIN_E15	Output	DIFF_DIR2
DirCtrl[3]	PIN_P14	Output	DIFF_DIR3
DirCtrl[4]	PIN_C15	Output	DIFF_DIR4
DirCtrl[5]	PIN_N8	Output	DIFF_DIR5
DirCtrl[6]	PIN_D4	Output	DIG_DIR_BANK1
DirCtrl[7]	PIN_C14	Output	DIG_DIR_BANK2
DirCtrl[8]	PIN_B14	Output	DIG_DIR_BANK3
DirCtrl[9]	PIN_L15	Output	DIG_DIR_BANK4
DirCtrl[10]	PIN_H13	Output	DIG_DIR_BANK5
DirCtrl[11]	PIN_N11	Output	DIG_DIR_BANK6
ExtClock	PIN_B8	Input	EXTCLK
GLOBAL_DIO18	PIN_J16	Input	DIG_D18
GLOBAL_DIO22	PIN_J15	Input	DIG_D22
GLOBAL_DIO46	PIN_H16	Input	DIG_D46
SRAM INTERFACE			
nBHE_RAM	PIN_C11	Output	NBHE_RAM
nBLE_RAM	PIN_P11	Output	NBLE_RAM
nOE_RAM	PIN_D11	Output	NOE_RAM
nWE_RAM	PIN_P4	Output	NEW_RAM
RAMa[0]	PIN_N4	Output	RAMA0
RAMa[1]	PIN_P2	Output	RAMA1
RAMa[2]	PIN_C12	Output	RAMA2
RAMa[3]	PIN_B13	Output	RAMA3
RAMa[4]	PIN_T11	Output	RAMA4
RAMa[5]	PIN_E14	Output	RAMA5
RAMa[6]	PIN_E16	Output	RAMA6
RAMa[7]	PIN_A13	Output	RAMA7
RAMa[8]	PIN_E13	Output	RAMA8
RAMa[9]	PIN_N1	Output	RAMA9
RAMa[10]	PIN_P1	Output	RAMA10
RAMa[11]	PIN_D6	Output	RAMA11
RAMa[12]	PIN_M4	Output	RAMA12
RAMa[13]	PIN_C4	Output	RAMA13
RAMa[14]	PIN_B10	Output	RAMA14

PIN ASSIGNMENTS

The SRAM Interface pins provide the interconnect between the Cyclone II device and the SRAM. Refer to the SRAM data sheet for more information. The data sheet is available from the manufacturer's web site listed immediately following the Table of Contents.

PIN ASSIGNMENTS

The IP Interface pins provide the interconnect between the Cyclone II device and the IP bus. Note that each IP line is buffered since the FPGA is not 5V tolerant. Refer to the IP Specifications available from **www.vita.com** for further information.

RAMa[15]	PIN_M1	Output	RAMA15
RAMd[0]	PIN_B7	Bidir	RAMD0
RAMd[1]	PIN_N6	Bidir	RAMD1
RAMd[2]	PIN_T10	Bidir	RAMD2
VHDL Name	**Pin #**	**Direction**	**Schematic Connection**
SRAM INTERFACE			
RAMd[3]	PIN_T6	Bidir	RAMD3
RAMd[4]	PIN_P6	Bidir	RAMD4
RAMd[5]	PIN_T8	Bidir	RAMD5
RAMd[6]	PIN_R4	Bidir	RAMD6
RAMd[7]	PIN_N3	Bidir	RAMD7
RAMd[8]	PIN_A10	Bidir	RAMD8
RAMd[9]	PIN_M3	Bidir	RAMD9
RAMd[10]	PIN_N2	Bidir	RAMD10
RAMd[11]	PIN_R10	Bidir	RAMD11
RAMd[12]	PIN_T7	Bidir	RAMD12
RAMd[13]	PIN_R5	Bidir	RAMD13
RAMd[14]	PIN_R11	Bidir	RAMD14
RAMd[15]	PIN_L14	Bidir	RAMD15
IP INTERFACE			
A[1]	PIN_N9	Input	FPGA_A1
A[2]	PIN_J2	Input	FPGA_A2
A[3]	PIN_J1	Input	FPGA_A3
A[4]	PIN_A9	Input	FPGA_A4
A[5]	PIN_D1	Input	FPGA_A5
A[6]	PIN_T9	Input	FPGA_A6
ACK_n	PIN_D10	Output	~FPGA_ACK
BS0_n	PIN_H1	Input	~FPGA_BS0
BS1_n	PIN_K1	Input	~FPGA_BS1
CLK8MZ	PIN_H2	Input	FPGA_CLK8MZ
DHIGH[0]	PIN_E2	Bidir	FPGA_D8
DHIGH[1]	PIN_C2	Bidir	FPGA_D9
DHIGH[2]	PIN_F3	Bidir	FPGA_D10
DHIGH[3]	PIN_E1	Bidir	FPGA_D11
DHIGH[4]	PIN_K4	Bidir	FPGA_D12
DHIGH[5]	PIN_C5	Bidir	FPGA_D13
DHIGH[6]	PIN_L2	Bidir	FPGA_D14
DHIGH[7]	PIN_J4	Bidir	FPGA_D15
DLOW[0]	PIN_L1	Bidir	FPGA_D0
DLOW[1]	PIN_A3	Bidir	FPGA_D1
DLOW[2]	PIN_A7	Bidir	FPGA_D2
DLOW[3]	PIN_G4	Bidir	FPGA_D3
DLOW[4]	PIN_D7	Bidir	FPGA_D4
DLOW[5]	PIN_P3	Bidir	FPGA_D5
DLOW[6]	PIN_K2	Bidir	FPGA_D6
DLOW[7]	PIN_E3	Bidir	FPGA_D7
DMAAck_n	PIN_L4	Input	~FPGA_DMAACK
DMAend_n	PIN_B4	Output	~FPGA_DMAEND

DMAReq0_n	PIN_C6	Output	~FPGA_DMAREQ0
DMAReq1_n	PIN_B6	Output	~FPGA_DMAREQ1
IDSEL_n	PIN_A4	Input	~FPGA_IDSEL
INTREQ0_n	PIN_A6	Output	~FPGA_INTREQ0
VHDL Name	**Pin #**	**Direction**	**Schematic Connection**
IP INTERFACE			
INTREQ1_n	PIN_A5	Output	~FPGA_INTREQ1
INTSEL_n	PIN_D2	Input	~FPGA_INTSEL
IOSEL_n	PIN_B9	Input	~FPGA_IOSEL
MEMSEL_n	PIN_D3	Input	~FPGA_MEMSEL
R_W_n	PIN_N10	Input	FPGA_R/~W
RESET_n	PIN_A8	Input	FPGA_~RST
Strobe_n	PIN_E4	Bidir	~FPGA_STROBE
PROGRAMMABLE CLOCK GENERATOR INTERFACE			
CLKOUT_REF	PIN_T3	Output	REFCLK
GEN_CLK	PIN_R9	Input	GENCLK
SCK	PIN_R12	Output	SCLK
SER_DATA	PIN_R3	Output	SDATA
CPLD CONTROL SIGNALS			
EnableCPLD	PIN_L3	Output	ENABLECPLD
IPRead_En_High	PIN_A11	Output	READ_ENABLE_H
IPRead_En_Low	PIN_B5	Output	READ_ENABLE_L
Strobe_Dir	PIN_N13	Output	INIT_DONE[1]

PIN ASSIGNMENTS

The Programmable Clock Generator Interface provides the interconnect between the Cypress clock generator and the Cyclone II device. Refer to the manufacturer's data sheet for further information.

The CPLD Control Signals are required signals to assist the CPLD in controlling the IP-EP2 module. Refer to the *Required VHDL* section earlier in this manual for further information.

1. The Init_Done signal was not used within the CPLD. As such, its function was changed to a direction control signal for the IP Bus Strobe_n signal.

Configuration Pins Table

The configuration pins do not have to be assigned in the Quartus II software. The table is for reference purposes only.

VHDL Name	Pin #	Schematic Connection
ASDO	PIN_C3	~ADSI
CONF_DONE	PIN_L13	CONFIG_DONE
DATA0	PIN_F1	DATA0
DCLK	PIN_H4	DCLK
MSEL0	PIN_J13	MSEL
MSEL1	PIN_K12	GND
nCE	PIN_G5	GND
nCEO	PIN_N14	Not Used
nCSO	PIN_F4	~CS
nStatus	PIN_M13	~STATUS
TCK	PIN_F2	FPGA_TCK
TDI	PIN_H5	FPGA_TDI
TDO	PIN_G2	FPGA_TDO
TMS	PIN_G1	FPGA_TMS

Power Pins

The remaining pins on the package are either unused, power, or ground pins. All GND and GND_PLL pins are connected to an internal ground plane in the PCB. All VCCIO pins are connected to 3.3V. All VCCINT, VCCD, and VCCA pins are connected to 1.2V. The unused pins are either left unconnected or connected to ground. Refer to the schematic provided in the EDK for further information. A complete list of pinouts for the Cyclone II FPGA is provided on the EDK CD in an Excel spreadsheet.

WEAK PULL-UP ASSIGNMENTS

Several I/O lines require that the weak pull-up resistors logic option be enabled in the FPGA. The pull-up resistors are required to prevent the I/O from floating. These assignments are done in the Assignment Editor. The following I/O pins require weak pull-ups: DIO[0] through DIO[47], DirCtrl[6], DirCtrl[7], DirCtrl[8], DirCtrl[9], DirCtrl[10], and DirCtrl[11].

DEVICE SETTINGS

The IP-EP2 module design requires that some FPGA device settings be fixed. These settings are found under *Device* in the *Assignments* category. Assume that all settings listed below are required and ENABLED (checked or selected) unless stated otherwise. Any disabled (unchecked) options are not listed.

Family: Cyclone II
Specific Device: EP2C20F256C8

Click the *Device & Pin Options* button for the follow settings. Remember that only ENABLED options are listed.

General Tab
- Auto-restart configuration after error
- Auto usercode (User can enter own usercode if desired.)

Configuration Tab
- Active Serial Configuration Scheme
- Use Configuration Device: EPCS4
- Generate Compressed Bitstreams (Optional)

Programming Files Tab
- Hexadecimal Output File
- Start Address: 0
- Count: UP

Unused Pin Tab
- Reserve as inputs tri-stated with weak pull-up

Dual Purpose Pins
- nCE0: Use as programming pin

Voltage Tab
- I/O standard: LVTTL

REVISION INFORMATION

REV	Date	Description
A	12/11/06	Initial Release
B	06/25/07	Added Revision Information. Modified Strobe_n to Bidir and added CPLD control signal Strobe_Dir in table on page 24. Removed Enable Init_Done requirement on pages 5 and 25. Added Frequently Asked Questions.
C	1/04/11	Correct EnableCPLD pin reference from F3 to L3 on pages 9 and 12. Updated references to Rev. C. EDK.
D	6/17/13	Updated to be compatible with Altera Quartus II Version 12.1 SP1 development tools